Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для компьютерного клуба»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы и программирование»
Воронин Никита Валерьевич
подпись: _____

Проверил:
преподаватель дисциплины,
преподаватель ЧУПО «ВШП»,
к.ф.н. Ткачев П.С.
оценка: _____
подпись: _____

Тверь, 2024 г.

Оглавление

Введение

Актуальность

Сейчас игровая индустрия стремительно развивается и не все люди в силу некоторых личных обстоятельств могут позволить себе дорогие комплектующие или консоли, по этому они, время от времени, посещают компьютерные клубы. Данная тема может считаться актуальной и достаточно интересной, по скольку многие процессы компьютерных клубов можно переложить также на другой бизнес или другие концепции.

Определение цели работы

Для начала следует внести некоторую ясность в тему и сформулировать цель курсовой работы. Тема курсовой работы “Разработка базы данных для компьютерного клуба”. Соответственно, цель работы – разработать базу данных, отражающую некоторые бизнес процессы, для условного компьютерного клуба. Прежде чем перейти к формулировке задач, необходимых для достижения поставленной цели, стоит кратко изложить что из себя вообще представляет “компьютерный клуб” как сущность или явление.

По версии сайта telegra.ph[5], компьютерные клубы (также известные как интернет-кафе или киберкафе) – это заведение предоставляющее доступ к Интернету. Это публичное заведение, где люди могут приходить работать, играть в онлайн-игры, общаться в социальных сетях и просто проводить время в виртуальном мире. Внутри интернет-кафе обычно имеются несколько компьютерных столов, каждый со своей клавиатурой, мышью и монитором. Кроме того, в заведении часто бывают доступны Wi-Fi, принтеры и сканеры. Из этого определения уже можно понять что-то о возможных бизнес процессах в условном компьютерном клубе, но это лучше оставить для основного раздела работы.

Краткая история возникновения феномена в России

В России первые компьютерные клубы появились в 1996 году: легендарные «Орки» в Москве и «Виртус» в Санкт-Петербурге. Редкие везунчики, у которых были компьютеры, ходили друг к другу с собственными машинами, чтобы поиграть по локалке. Все остальные тусовались в игровых клубах.

“Только откроется клуб на районе, как в школе уже говорят об этом: «Ну че, ходил в новые компы?» Мы с друзьями неделю экономили на обедах, чтобы на выходных посидеть в компах подольше. Часто приходишь в компы, стучишь в дверь и через решетку спрашиваешь у админа: «Есть свободные?» Он отвечает, что через два часа освободятся два компа. Забиваешь очередь и уходишь ждать. Люди стояли на морозе, просили пустить в тамбур погреться или в зал — посмотреть за игрой других.

Еще бывало так: на ночь собиралась компания, админ запирал их всех внутри и уходил домой спать. Жуткая вещь: на окнах решетки, телефонов нет никаких, если вдруг пожар — хана.”

Алексей Офицеров, заядлый посетитель клубов в 90-х [3]

Современные же компьютерные клубы отличаются следующими особенностями:

1. Мощное оборудование. Современные клубы устанавливают сетевое оборудование и игровые системы, которые полностью удовлетворяют потребности популярных проектов и имеют запас мощности на несколько лет вперёд.
2. Пополнение списка актуальных игр. Современные клубы следят за выходом дополнений, оперативной установкой патчей и фиксов.
3. Удобство клиентов. Клубы предоставляют эргономичные девайсы для игровой зоны, а также разрешают приходить со своими.

4. Сервис. Современные клубы стремятся к безупречному сервису, например, имеют удобные парковочные места, комнаты для отдыха и lounge-зоны.

Разница между “было” и “стало” вполне очевидна.

Постановка задач

Исходя из определения выше – сформулируем возможные задачи для достижения цели курсовой работы:

1. Определение бизнес процессов условного компьютерного клуба.
2. Сформулировать требования к разрабатываемой базе данных
3. Выбор СУБД(системы управления базами данных) для реализации базы данных.
4. Построение схемы базы данных.
5. Разработать набор связанных таблиц базы данных, исходя из требований к базе данных и бизнес процессов условного компьютерного клуба.
6. Заполнить таблицы тестовыми данными.
7. Реализовать бизнес процессы на уровне разработанной базы данных средствами выбранной СУБД.
8. Провести тестирование реализованных процессов тестовыми данными.

Объект исследования

Объектом исследования является процесс разработки баз данных, предназначенной для обслуживания бизнес процессов условного компьютерного клуба.

Метод исследования

Моделирование, а именно моделирование процессов внутри компьютерного клуба и пользовательского поведения.

Первая глава

Определение бизнес процессов условного компьютерного клуба

Для наглядности нужно дать определение “бизнес процессу”, как понятию. Сам по себе бизнес процесс — совокупность взаимосвязанных мероприятий или работ, направленных на создание определённого продукта или услуги для потребителей. [1]

Выделяют три вида бизнес-процессов:

1. управляющие — бизнес-процессы, которые управляют функционированием системы, такие как корпоративное управление и стратегический менеджмент;
2. операционные — представляющие основную деятельность организации, создающие основной поток доходов (снабжение, производство, маркетинг, продажи или взыскание долгов);
3. поддерживающие — обслуживающие организацию (бухгалтерский учет, подбор персонала, техническая поддержка).

Бизнес-процесс начинается со спроса потребителя и заканчивается его удовлетворением. Процессно-ориентированные организации стараются устранять барьеры и задержки, возникающие на стыке двух различных подразделений организации при выполнении одного бизнес-процесса.

Бизнес-процесс может быть декомпозирован на несколько подпроцессов, процедур и функций, которые имеют собственные атрибуты, однако также направлены на достижение цели основного бизнес-процесса. Такой анализ бизнес-процессов обычно включает в себя составление карты бизнес-процесса и его подпроцессов, разнесенных между определёнными уровнями активности.

Бизнес-процессы должны быть построены таким образом, чтобы создавать стоимость и ценность для потребителей и исключать любые необязательные или

вовсе лишние активности. На выходе правильно построенных бизнес-процессов увеличиваются ценность для потребителя и рентабельность (меньшая себестоимость производства товара или услуги).

Бизнес-процессы могут подвергаться различному анализу в зависимости от целей моделирования. Анализ бизнес-процессов может применяться при бизнес-моделировании, функционально-стоимостном анализе, формировании организационной структуры, реинжиниринге бизнес-процессов, автоматизации технологических процессов.

В данном случае условный компьютерный клуб, назовем его “Игрульки”, является перспективно масштабируемым бизнесом с возможностью расширения. Относительно этого процессы внутри компьютерного клуба должны быть универсальными для всех филиалов организации, быстро и точно обрабатывать данные пользователей и сотрудников, работать надежно и бесперебойно. В основе клуб должен приносить прибыль за счет посетителей, которые пополняют кошелек на своем аккаунте для запуска сессии на выбранном компьютере.

Тезисно представленные в виде списка искомые бизнес процессы:

1. Хранение и обслуживание пользовательских данных клиентов и сотрудников клуба “Игрульки”
2. Реализация функционала пополнения баланса пользовательского аккаунта.
3. Активация сессий пользователя за н-ным компьютером в н-ном зале при н-ном сотруднике на смене и достаточном балансе.
4. Сохранение и отображение данных о совершенных пополнениях.

Необходимые основные процессы сформулированы и оглашены. Исходя из этого можно перейти к формулировке требований к базе данных.

Формулировка требований к разрабатываемой базе данных

Какие вообще бывают требования к базам данных?

Согласно источнику[2], существуют следующие виды бизнес-требований:

1. Функциональные. Описывают конкретные функции системы или проекта, которые больше всего важны для бизнеса. Отвечают на вопрос «Что должна делать система?».

2. Нефункциональные. Описывают характеристики продукта, которые определяют, как он должен выполнять свои функции. Отвечают на вопрос «Как должна работать функция системы?».

Бизнес-требования помогают:

1. зафиксировать требования заказчика проекта;
2. собрать потребности клиентов;
3. команде разработчиков и всем сотрудникам понять, какой конечный продукт нужно получить;

Исходя из изложенного выше можно выделить несколько требований:

1. База данных должна быть реляционной, по скольку в реляционной базе данные организованы в таблицы, содержащие информацию о каждом объекте и представляющие заранее определённые категории через строки и столбцы. Такое структурирование данных делает доступ к ним эффективным и гибким (Что такое реляционные базы данных [4])
2. База данных должна содержать таблицы для организации работы сотрудников, пользовательского интерфейса и некоторой логики клуба ”Игрульки”
3. Таблицы должны быть связаны между собой для организации логики бизнес процессов и корректного отображения данных.

4. База данных должна поддрерживать большое количество асинхронных запросов со стороны клиентов сервиса, следовательно, должна быть легко внедряемой и универсальной.

Подробнее поговорим о реляционных базах данных и какие им бывают альтернативы:

Реляционные базы данных предназначены для поддержания согласованности и целостности данных, а также обеспечения связей и ограничений между различными таблицами. Они полагаются на язык структурированных запросов (SQL) для запроса, манипулирования и организации данных.

Альтернатива реляционным базам данных — нереляционные базы данных (базы данных NoSQL). Они предназначены для хранения данных в форматах, отличных от таблиц, и предоставляют более простое, гибкое и масштабируемое решение для хранения и управления неструктурированными или полуструктурированными данными.

Некоторые широко используемые нереляционные базы данных включают:

— MongoDB,

— Cassandra,

— Redis,

— Neo4j.

Однако, данные СУБД не совсем удовлетворяют все критерии для реализации сервиса условного компьютерного клуба “Игрульки”, по этому от них в последствии придется отказаться.

Основные требования к базе данных, на мой взгляд, сформулированы. Исходя из этих требований можно перейти к следующей задаче.

Выбор СУБД для реализации базы данных

В качестве системы управления базами данных была выбрана MySQL. Данная СУБД является реляционной, достаточно производительной и функциональной для выполнения поставленных выше задач и требований, а именно:

1. MySQL имеет в своем функционале различные функции безопасности, включая возможность установки привилегий пользователя, шифрование данных, аутентификацию и аудит, чтобы обезопасить информацию, хранящуюся в базах данных. Поддерживает множество языков программирования, таких как: Python, C/C++, Java, JavaScript, Go, Delphi, Erlang и др.
2. MySQL обладает хорошей производительностью и быстродействием благодаря оптимизированным алгоритмам выполнения запросов. Умеет работать с большими объемами данных с минимальными задержками. Легко масштабируется и подходит для больших баз данных.
3. MySQL может использоваться как для небольших веб-приложений, так и для серьезных корпоративных систем. СУБД предлагает различные методы масштабирования, включая горизонтальное и вертикальное масштабирование, что дает возможность расширять базы данных при возникновении необходимости. Это позволяет обрабатывать большое количество одновременных запросов и поддерживать беспрепятственный доступ к данным для пользователей.
4. В MySQL реализованы различные типы данных, индексы, хранимые процедуры, триггеры и другие функции, обеспечивающие гибкость при обработке данных. Это дает возможность создания сложных схем данных. MySQL совместима с довольно большим количеством операционных системам.
5. Бесплатность и открытые исходники: рассматриваемый продукт имеет открытый исходный код и может быть использован бесплатно. Это делает его доступным для большого круга разработчиков и пользователей.

6. MySQL имеет большое сообщество разработчиков и пользователей, что дает возможность доступа к различным ресурсам и помощи при возникновении проблем.

Перечисленные возможности как нельзя лучше подходят для реализации подобной базы данных. Однако, ниже приведены некоторые недостатки MySQL, на которые стоит обратить внимание:

1. MySQL может иметь проблемы с производительностью при обработке больших объемов данных или когда требуется обработка сложных запросов или подключение большого количества клиентов. Это может привести к замедлению работы или даже отказу в обслуживании.
2. MySQL имеет ограниченный набор типов данных по сравнению с некоторыми другими СУБД. Например, поддержка временных данных и географических данных ограничена по сравнению с некоторыми другими СУБД.
3. MySQL может столкнуться с ограничениями в масштабируемости и доступности при взаимодействии с большим количеством клиентов. Тогда отказоустойчивость и возможность горизонтального масштабирования могут быть ограничены.
4. MySQL может быть сложным для администрирования, особенно для новичков. Настройка и оптимизация параметров конфигурации может быть сложной задачей, требующей глубоких знаний и опыта.
5. MySQL может иметь уязвимости в сфере безопасности, например, возможность атаки SQL-инъекцией или недостаточная защита данных. Это требует дополнительных мер безопасности для уверенной надежности и защиты баз данных.

Также, для большей информативности нужно выделить области применения MySQL:

1. MySQL часто применяется для построения динамических сайтов и интернет-приложений. При этом СУБД используется в сочетании с веб-серверами (Apache, Nginx)
2. Многие предприятия организации используют MySQL для создания и управления баз(ами) данных своих бизнес-приложений. Данная СУБД дает возможность надежно хранить и легко получать доступ к данным, что позволяет эффективно работать с клиентами, заказами, продуктами и другими составляющими бизнеса, получая информацию о них из корпоративных баз данных.
3. Базы MySQL используются как хранилище данных для аналитических и отчетных систем. С помощью рассматриваемого программного продукта можно хранить большие объемы данных, а также выполнять сложные запросы и агрегировать данные, чтобы проводить анализ и создавать отчеты.
4. MySQL используется также как СУБД для мобильных приложений, хранящих различные типы данных, такие как пользовательские профили, настройки, результаты и др. Во многом из-за своей надежности и хорошей производительности MySQL хорошо подходит для решения данных задач.
5. MySQL может быть встроен в другие приложения и устройства для взаимодействия с небольшими базами данных. Это могут быть системы управления контентом (CMS), блоги, форумы, физические устройства, такие как маршрутизаторы и т. д.

Почему MySQL а не PostGres ?

Выбор между MySQL и PostgreSQL зависит от конкретных потребностей и приоритетов бизнеса. Обе СУБД имеют свои сильные стороны и особенности. В контексте компьютерного клуба, MySQL может быть предпочтительным по следующим причинам:

1. Легкая установка и настройка, что особенно важно для небольших команд или тех, кто не имеет глубокого опыта в администрировании баз данных.
2. Менее сложный процесс конфигурации по сравнению с PostgreSQL.
3. Обычно MySQL превосходит PostgreSQL в операциях на чтение, что может быть важным для приложений, где чтение данных происходит чаще, чем запись.
4. MySQL широко поддерживается многими хостинг-провайдерами и платформами.
5. Легкая интеграция MySQL с популярными веб-приложениями, такими как WordPress, Joomla и другие, что может быть полезно для создания веб-сайта клуба или онлайн-сервисов.
6. Более интуитивно понятный интерфейс и множество инструментов для администрирования, таких как phpMyAdmin.
7. Оптимизирован для работы с веб-приложениями и часто используется в популярных веб-технологиях.
8. Высокая производительность в типичных веб-нагрузках, что важно для веб-сайтов компьютерного клуба.
9. Обычно MySQL требует меньше усилий на администрирование и техническую поддержку, что может быть важным фактором для небольших организаций с ограниченными ресурсами.
10. Большое количество специалистов, знакомых с MySQL, что облегчает поиск сотрудников для технической поддержки и развития.

Примеры использования в компьютерном клубе

1. Регистрация и учет клиентов: Быстрая обработка регистрационных данных и хранение истории посещений клиентов.
2. Управление оборудованием: Учет состояния и использования компьютеров и периферийных устройств.
3. Бронирование и резервирование: Обработка запросов на бронирование времени на компьютерах.

4. Финансовые операции: Ведение учета оплат за услуги и управление абонементами.
5. Обслуживание и поддержка: Хранение данных о техническом обслуживании и поддержке пользователей.

Выбор MySQL может быть оправдан для компьютерного клуба, учитывая простоту использования, высокую производительность на чтение, широкую поддержку и интеграцию с веб-приложениями. MySQL может обеспечить эффективное управление данными с минимальными затратами на установку и администрирование, что особенно важно для небольших и средних предприятий. Однако, окончательный выбор должен базироваться на конкретных потребностях и характеристиках бизнеса.

Краткий вывод после проведения анализа возможных решений и определения бизнес процессов условного компьютерного клуба “Игрульки”

Были выявлены все необходимые критерии для работы с бизнес процессами условного компьютерного клуба “Игрульки”. Основной упор в выборе СУБД был сделан на реляционные базы данных и СУБД MySQL. Выполнены задачи по анализу процессов и возможных решений на уровне СУБД.

Собранные данные позволяют приступить к разработке искомой базы данных с учетом всех нюансов деятельности условного компьютерного клуба “Игрульки”.

Список процессов которые нужно реализовать:

1. Регистрация и учет клиентов.
2. Управление оборудованием клуба.
3. Финансовые операции с балансом клиентов.
4. Сбор данных о пользователях и сотрудниках в рамках одной транзакции и/или сессии.

Вторая глава

Построение схемы базы данных

Перед началом работы с СУБД был выбран официальный инструмент для работы с MySQL – MySQL Workbench.

MySQL Workbench - это официальная система для разработки и администрирования баз данных. Она предлагает множество функций и возможностей для управления данными. Преимущества MySQL Workbench включают возможность хранить большие объемы данных, отличные возможности управления данными, высокую производительность, кроссплатформенность, бесплатность, выбор движков и безопасность.

Возможности и использование MySQL Workbench

1. Проектирование базы данных

Workbench предлагает широкий спектр функций для проектирования и моделирования, включая создание сложных ER-моделей и возможность выполнять обратное и прямое проектирование. Кроме того, администраторы, разработчики и архитекторы могут легко вносить изменения и управлять документами для проектирования своей базы данных.

2. Разработка

Пользователи могут создавать и оптимизировать SQL-запросы, используя визуальные инструменты, предоставляемые MySQL Workbench, что позволяет им эффективно выполнять запросы. Другие функции, которые помогают и упрощают задачу разработки и выполнения запросов, включают автозаполнение, выделение синтаксиса разными цветами, предоставление истории выполнения запросов и повторное использование фрагментов SQL. Панель подключения может хранить различные подключения к базе данных и управлять ими для баз данных, включая MySQL fabric. Обозреватель объектов в MySQL Workbench обеспечивает мгновенный доступ к схеме и объектам базы данных.

3. Администрирование

Визуальная консоль предоставляется в MySQL workbench; администраторы баз данных и разработчики могут использовать ее для просмотра всей среды базы данных. Другие доступные инструменты могут оказаться полезными при настройке сервера, администрировании пользователей, аудите данных для проверки, проверке работоспособности базы данных и резервного копирования, а также восстановлении данных. Визуальные инструменты MySQL Workbench облегчают все эти задачи.

ER диаграмма и набор таблиц проекта

ER-диаграмма – это разновидность блок-схемы, на которой показано, как разные сущности (люди, объекты, концепции и так далее) связаны между собой внутри системы.

ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

Перед построением ER-диаграммы определим 6 таблиц, в которые будут входить перечисленные в прошлой главе возможности:

1. ‘users’ – данная таблица отражает сущность пользователя, т.е. клиента условного компьютерного клуба. Таблица должна содержать данные для успешной авторизации пользователя, сведения о балансе пользователя, каких либо данных для подтверждения пасспорта.
2. ‘employees’ – данная таблица отражает сущность сотрудника условного компьютерного клуба. Таблица должна содержать данные для успешной авторизации сотрудника, сведения о его полном имени, описание для оставления заметок, данные о заработке сотрудника в час.
3. ‘rooms’ – данная таблица отражает сущность комнаты условного компьютерного клуба. Таблица должна содержать данные о комнате, ее

заголовок, описание и статус(VIP или нет), по скольку статус VIP и стоимость определяется именно комфортом помещения.

4. ‘computers’ – данная таблица отражает сущность конкретного компьютера в конкретной комнате, соотвественно должна быть связана с сущностью комнаты, при этом в одной комнате может быть несколько компьютером, но один компьютер не может быть в нескольких комнатах одновременно.
5. ‘payments’ – данная таблица отражает сущность факта оплаты. Она должна быть связана с таблицами сущностей пользователя и сотрудника для возможности отслеживания того какой пользователь совершил пополнение в смену какого сотрудника. Один пользователь может совершить несколько оплат, но у одной оплаты не может быть несколько пользователей одновременно.
6. ‘sessions’ – эта таблица отражает сущность сессии конкретного пользователя за конкретным компьютером в смену конкретного сотрудника. Она должна содержать данные о начале сессии, ее конце, дате начала, и иметь связь с таблицами сущностей пользовтелей, сотрудников и компьютеров. При этом, у одного пользователя, сотрудника и компьютера может быть несколько сессий.

Используя все формулировки сущностей и их таблиц выше, приступим к разработке ER-диаграммы. (Через инструмент построения диаграмм MySQL Workbench)

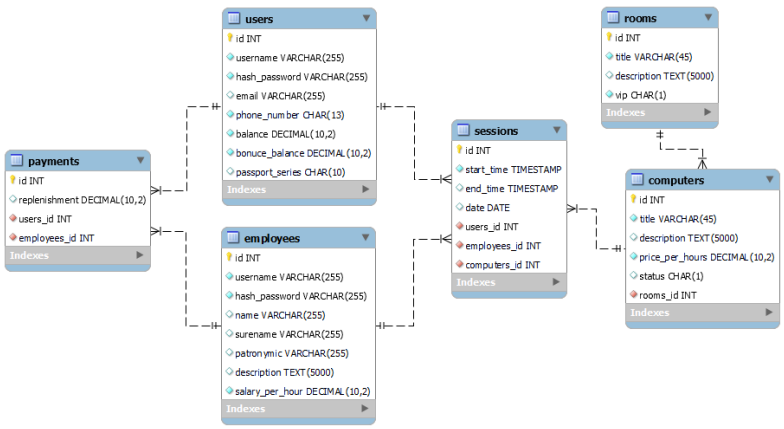


Рисунок 1

Таблица users

Расположим таблицу users на диаграмме (Рис.1) и зарегистрируем все необходимые колонки для сущности пользователя.

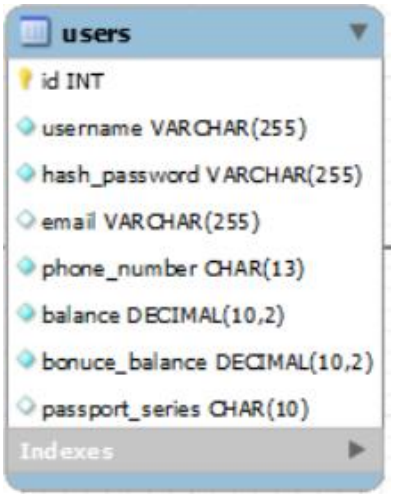


Рисунок 2

Таблица содержит поля: username, hash_password, email, phone_number, balance, bonuce_balance, passport_series.

1. Для поля id выбран тип данных INT, по скольку он является первичным ключом таблицы и должен быть авто инкрементируемым.
2. Для поля username выбран тип данных VARCHAR(255), по скольку оно будет хранить имя пользователя, которое может состоять из разных символов, в том числе и специальных.
3. Для поля hash_password выбран тип данных VARCHAR(255), по скольку оно будет хранить хэшированный пароль пользователя, который, обычно, всегда имеет фиксированную длину
4. Для поля email выбран тип данных VARCHAR(255), по аналогии с полем username, по скольку email также является строкой содержащей специальные символы.
5. Поле phone_number должно содержать номер телефона, который в РФ имеет, пока что, фиксированную длину в 13 символов. От сюда и выбор типа данных CHAR(10)
6. Поле passport_series содержит серию и номер паспорта пользователя. Серия и номер пасспорта гражданина РФ могут состоять максимум из 10 символов

Отдельного внимания стоят поля ‘balance’ и ‘bonuce_balance’, они нужны для хранения, обработки и отображения баланса пользователя. При этом бонусный баланс предусматривает возможность реализации функционала “кэшбжка” или системы подарков и наград, что, в теории, должно подогревать интерес игроков к этой “механике”, по скольку “геймеры любят цифры”.

Для обработки любого денежного баланса следует использовать тип данных DECIMAL, в данном случае DECIMAL(10, 2), т.е. десять знаков до запятой и два после. В последсвии каждое поле, так или иначе содержащее баланс или что-то связанное с деньгами будет иметь тип данных DECIMAL(10, 2).

Таблица employees

Разумно сначала создавать таблицы не содержащие внешние ключи.

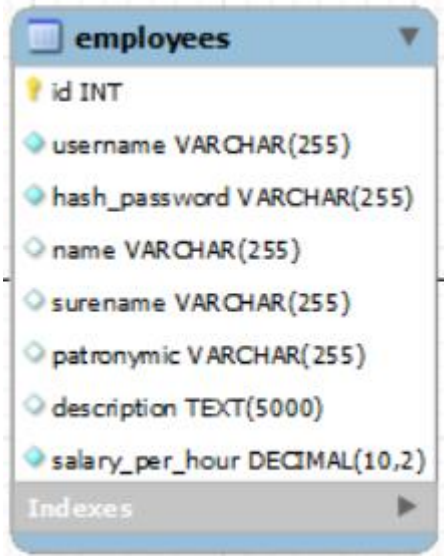


Рисунок 3

По аналогии с таблицей users (Рис.2) разместим таблицу employees, содержащую поля: id, username, hash_password, name, surename, patronymic, description, salary_per_second.

Не слишком хорошая практика разделять аккаунты пользователей по разным таблицам, но тут это обусловлено тем, что в условном компьютерном клубе админ может залогинить сессию без оплаты. Помимо этого таблица этой сущности хранит данные немного отличные от таблицы users, например – заработная плата в час.

Было бы странно хранить эти параметры в одной таблице, как минимум это не очень удобно для отображения данных.

Особенного внимания требует поле description и его тип данных DESCRIPTION(5000), в данном случае описание может хранить до пяти тысяч символов, в последующих таблицах с похожими полями также будет использоваться этот тип данных.

Таблица rooms

Как было сказано выше, данная таблица является сущностью комнаты условного компьютерного клуба, по этому она содержит заголовок, описание и данные о VIP статусе комнаты.

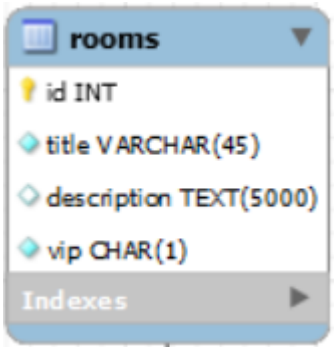


Рисунок 4

Как видно на Рис.4, таблица содержит заголовок с типом данных VARCHAR(45), по скольку заголовок не должен быть достаточно длинным, тут также присутвует текстовое описание с ограничением в 5000 символов. VIP статус комнаты отражается полем vip, которое может содержать всего один символ(CHAR(1)). Это может быть отличной альтернативой типам данных BOLEAN или ENUM, по скольку они немного меняют алгоритмы поиска по базе данных, что может понизить производительность базы при слишком большом количестве данных в таблицах.

Таблица computers

В этой таблице computers отражена сущность компьютера в конкретной комнате, по этому для связи появляется внешний ключ от таблицы rooms.

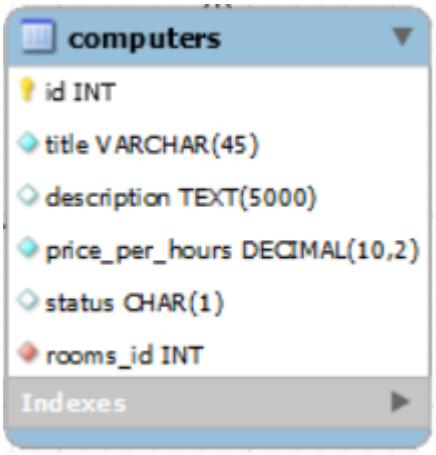


Рисунок 5

Также тут указана стоимость за один час, следовательно, можно будет с помощью операторов MYSQL редактировать стоимость всех компьютеров из комнаты с vip статусом. В данной таблице стоит обратить внимание на колонку status. Логически она похожа на колонку vip из таблицы rooms, но выполняет другую функцию, а именно – отражает доступность компьютера. (1-компьютер доступен, 0-компьютер недоступен)

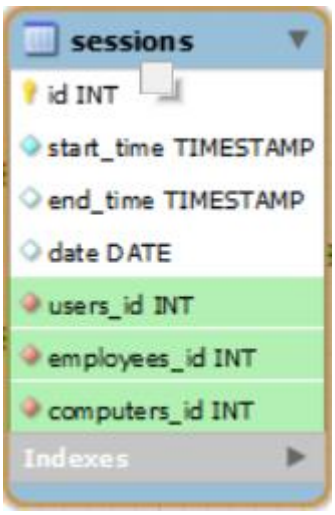


Рисунок 6

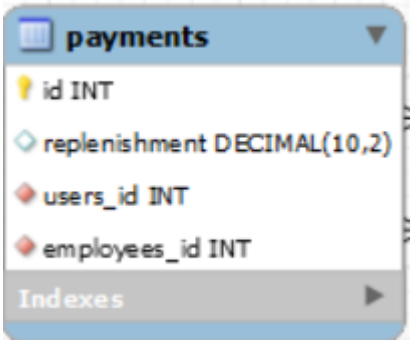


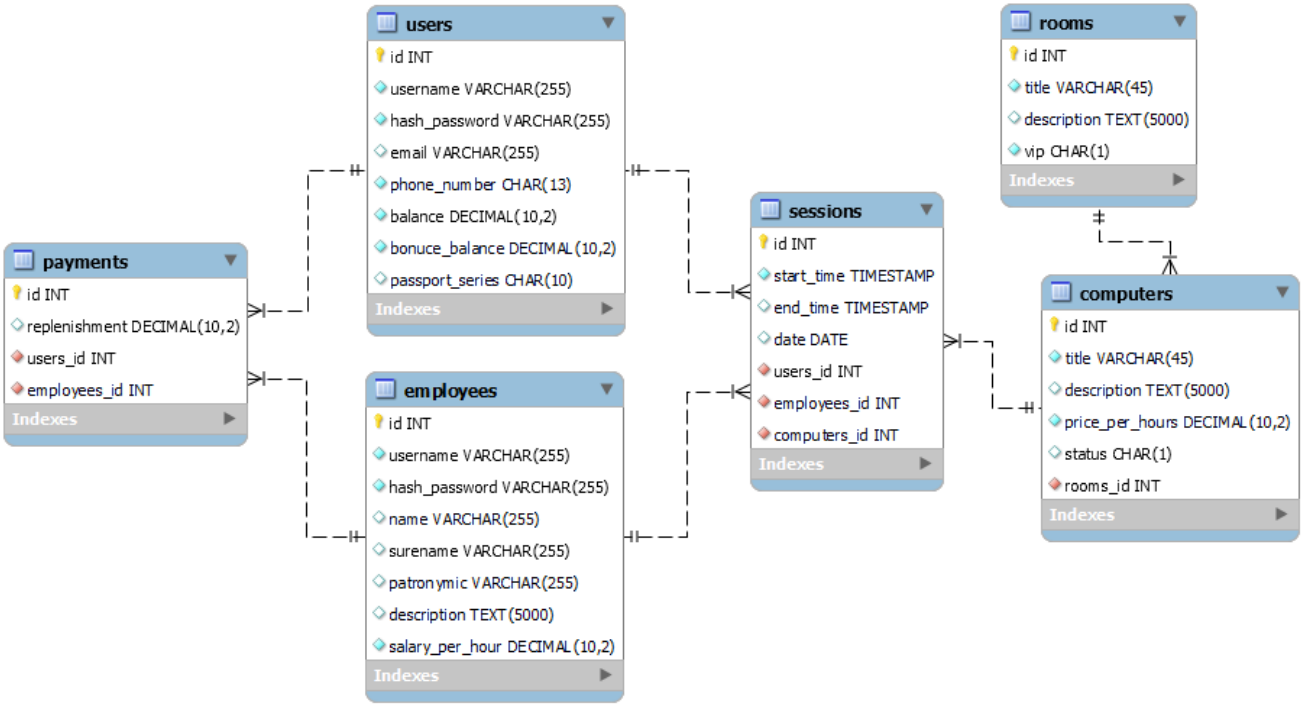
Рисунок 7

Таблицы sessions и payments для организации виртуального кошелька пользователя и основного функционала условного компьютерного клуба

Таблица sessions (Рис.6) содержит тип данных TEMESTAMP и дефолтным значением выставляет текущее время сервера в поле start_time. Поле end_time может не иметь значение и должно принимать его только к концу сессии. В идеале валидация поля должна происходить на уровне БД, например, чтобы не допустить добавления времени завершения сессии, которое будет раньше времени начала сессии. Однако, по скольку оно назначается автоматически после завершения сессии на уровне клиента конкретного компьютера, было решено отказаться от этой валидации вовсе.

Оплата сессии происходит не напрямую, а через баланс пользователя в личном кабинете, который, соответственно, нужно пополнить перед началом сессии. Для этого и существует таблица payments, которая получает первичные ключи сотрудника и пользователя, а также сумму оплаты.

В итоге все таблицы, исходя из перечисленного выше, имеют между такие связи:



Связи между таблицами

Краткое описание используемых в базе данных условного компьютерного клуба: Таблицы между собой связывает тип связи “один ко многим”, более подробно логика данных связей расписана в пункте “ ER диаграмма и набор таблиц проекта” на странице 15.

Я старался избегать связей “один к одному” и “многие ко многим”, поскольку, на мой субъективный взгляд, данные типы связей сильно усложняют разработку базы данных и заполнение таблиц тестовыми данными. Что, следовательно, ведет к проблемам с масштабируемостью и обслуживанием сервиса в будущем.

Разработка указанных выше таблиц.

Воспользуемся инструментом конвертации данных MySQL Workbench и сгенерируем таблицы из ER-диаграммы.

В данном случае вот так выглядит код для создания таблицы ‘users’

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NOT NULL,
  `hash_password` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NULL,
  `phone_number` CHAR(13) CHARACTER SET 'ascii' NOT NULL,
  `balance` DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.00,
  `bonuce_balance` DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.00,
  `passport_series` CHAR(10) NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE,
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE,
  UNIQUE INDEX `phone_number_UNIQUE` (`phone_number` ASC) VISIBLE,
  UNIQUE INDEX `passport_series_UNIQUE` (`passport_series` ASC)
VISIBLE)
```

Поле id является автоинкрементированным первичным ключом, username не может быть Null, hash_password также не может быть Null, по скольку пользователь без имени пользователя и пароля существовать не может чисто логически.

По аналогии создание таблицы ‘employees’

```
CREATE TABLE IF NOT EXISTS `employees` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NOT NULL,
  `hash_password` VARCHAR(255) NOT NULL,
  `name` VARCHAR(255) NULL,
  `surename` VARCHAR(255) NULL,
  `patronymic` VARCHAR(255) NULL,
  `description` TEXT(5000) NULL,
  `salary_per_hour` DECIMAL(10,2) NOT NULL DEFAULT 0.00,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE)
```

Таблицы ‘rooms’

```
CREATE TABLE IF NOT EXISTS `rooms` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT(5000) NULL,
  `vip` CHAR(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

Пример кода для всех таблиц слишком велик для указания тут (см. Приложение 1.)

Заполнение тестовыми данными

Для реализации запросов необходимо заполнить таблицу тестовыми данными, вот пример для тестовых данных пользователей из таблицы ‘users’

```
INSERT INTO users (`username`, `hash_password`, `email`,
`phone_number`, `balance`, `bonuce_balance`, `passport_series`)
VALUES
('user1', 'hash1', 'user1@example.com', '1234567890123', 100.00,
10.00, '4500123456'),
('user2', 'hash2', 'user2@example.com', '2234567890123', 200.00,
20.00, '4501234567'),
('user3', 'hash3', 'user3@example.com', '3234567890123', 300.00,
30.00, '4502345678'),
. . .
```

Код для заполнения данными всех таблиц (см. Приложение 2)

Реализация бизнес процессов на уровне СУБД

Для наглядности стоит огласить перечень инструментов MYSQL, подходящий для реализации бизнес процессов в искомой базе данных:

1. Типовые запросы:

   1.1 Классический запрос состоит из шести операторов:

   SELECT — выбирает отдельные столбцы или всю таблицу целиком (обязательный).

   FROM — указывает, из какой таблицы получить данные (обязательный).

   WHERE — условие, по которому SQL выбирает данные.

   GROUP BY — столбец, по которому данные будут группироваться.

   HAVING — условие, по которому сгруппированные данные будут отфильтрованы.

   ORDER BY — столбец, по которому данные будут отсортированы.

2. Триггеры:

   Триггер в MySQL — это определяемая пользователем SQL-команда, которая автоматически вызывается во время операций INSERT, DELETE или UPDATE. Код триггера связан с таблицей и уничтожается после удаления таблицы. Вы можете определить время действия триггера и указать, когда его нужно активировать – до или после определенного события базы данных.

3. Представления:

   В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе. Содержимое представления динамически вычисляется на основании данных, находящихся в реальных таблицах. Изменение данных в реальной таблице базы данных немедленно отражается в содержимом всех представлений, построенных на основании этой таблицы.

4. Транзакции:

   Это команды или блок команд (инструкций), которые успешно завершаются как единое целое, при этом в базе данных все внесенные изменения фиксируются на постоянной основе, или отменяются, т.е. все изменения,

внесенные любой командой, входящей в транзакцию, будут отменены. Другими словами, если одна команда или инструкция внутри транзакции завершилась с ошибкой, то все, что было отработано перед ней, также отменяется, даже если предыдущие команды завершились успешно.

По порядку рассмотрим примеры реализации каждого функционала базы данных:

1. Типовые запросы:

Этот запрос возвращает информацию о сессиях пользователей, включая данные о пользователях, сотрудниках и компьютерах

```
SELECT
    p.id AS payment_id,
    p.replenishment,
    p.users_id AS user_id,
    u.username AS user_username,
    u.email AS user_email,
    e.id AS employee_id,
    e.username AS employee_username
FROM
    payments p
JOIN
    users u ON p.users_id = u.id
JOIN
    employees e ON p.employees_id = e.id;
```

Этот запрос возвращает информацию о всех доступных компьютерах и их комнатах:

```
SELECT
    c.id AS computer_id,
    c.title AS computer_title,
    c.description AS computer_description,
    c.price_per_hours AS computer_price_per_hour,c.status,
    r.id AS room_id,
    r.title AS room_title,
    r.description AS room_description
FROM
    `computers` c
JOIN
    `rooms` r ON c.rooms_id = r.id
WHERE
    c.status = '1'; -- 1  означает, что компьютер доступен
```

Этот запрос возвращает информацию о платежах, включая данные о пользователях и сотрудниках, которые обработали эти платежи

```
SELECT
    p.id AS payment_id,
    p.replenishment,
    p.users_id AS user_id,
    u.username AS user_username,
    u.email AS user_email,
    e.id AS employee_id,
    e.username AS employee_username
FROM
    `payments` p
JOIN
    `users` u ON p.users_id = u.id
JOIN
    `employees` e ON p.employees_id = e.id;
```

Этот запрос возвращает информацию о сотрудниках и их месячных зарплатах, исходя из почасовой ставки и 160 рабочих часов в месяц:

```
SELECT
    e.id AS employee_id,
    e.username,
    e.name,
    e.surename,
    e.patronymic,
    e.salary_per_hour,
    (e.salary_per_hour * 160) AS monthly_salary -- Предполагаем 160 часов работы в месяц
FROM
    `employees` e;
```

Следующий запрос требует реализации и использования функции, по этому (см. Приложение 3)

2. Триггер:

Данный триггер реализует обновление баланса после факта пополнения в таблице payments от имени пользователя конкретного пользователя.

```
DELIMITER $$
CREATE TRIGGER after_payment_insert
AFTER INSERT ON `payments`
FOR EACH ROW
BEGIN
  -- Обновляем баланс пользователя
  UPDATE `users`
  SET balance = balance + NEW.replenishment
  WHERE id = NEW.users_id;
END$$
DELIMITER ;
```

3. Представление:

В данном случае представление реализует отображенние всех данных о всех сессиях в базе данных (см. Приложение 4)

4. Транзакции:

Данная транзакция также является хранимой процедурой. Она выполняет функцию пополнения баланса пользователя в таблице `users`, только если пополнение не превышает определенный лимит. (см. Приложение 5)

Также была реализована хранимая процедура, которая проверяет остаток на балансе пользователя и создает сессию. (см. Приложение 6)

Система ролей

В качестве настройки ролей были созданы два пользователя, отвечающих за выполнение разных операций с базой данных:

1. Пользователь admin

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin_password';
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;
```

2. Пользователь user

```
CREATE USER 'user'@'%' IDENTIFIED BY 'password';
-- Предоставить привилегии SELECT к таблице users
GRANT SELECT ON `mydb`.`users` TO 'user'@'%';
-- Предоставить привилегии SELECT к таблице sessions
GRANT SELECT ON `mydb`.`sessions` TO 'user'@'%';
-- Предоставить привилегии SELECT к таблице computers
GRANT SELECT ON `mydb`.`computers` TO 'user'@'%';
```

Вывод после реализации и тестирования искомой базы данных

В первую очередь можно убедиться, что MySQL как нельзя лучше подходит для реализации подобных задач в связи с наличием удобных инструментов установки, настройки, разработки и отладки таких баз данных. Но об этом подробнее в заключении.

Заключение

Общие выводы о работе с MySQL

1. Реляционные базы данных как нельзя лучше подходят для реализации бизнес процессов условного компьютерного клуба.
2. Разработка базы данных трудоемкий процесс, требующий больших усилий для построения грамотной логики работы и взаимодействия сущностей и их данных между собой.
3. Хорошая база данных – та, которая стабильно выполняет многие процессы сервиса без необходимости постоянно незащищённым способом обращаться к SQL инъекциям посредством других языков.

Достигнутые цели и задачи

Цель – разработать базу данных для компьютерного клуба. По мере продвижения к достижению этой цели был поставлен и выполнен ряд задач:

1. Изучен объект исследования посредством моделирования бизнес процессов условного компьютерного клуба “Игрульки”. А именно – изучен процесс разработки баз данных, предназначенных для обслуживания бизнес процессов условного компьютерного клуба.
2. Построена ER-диаграмма и спроектирован необходимый набор таблиц на основе определенных процессов и логики работы компьютерного клуба. Между таблицами были установлены взаимные связи, необходимые для корректной работы с сущностями и их данными.
3. Реализованы хранимые процедуры с обработками исключений, триггер, типовые запросы, функция представления и другой функционал базы данных.

На основе перечисленного выше могу сделать вывод, что цель курсовой работы по разработке базы данных для компьютерного клуба была достигнута.

Список используемой литературы

Электронные ресурсы

1. Википедия[Электронный ресурс]/Режим доступа: https://ru.wikipedia.org/wiki/Бизнес-процесс
2. Компас[Электронный ресурс]/Режим доступа: https://getcompass.ru/blog/posts/biznes-trebovaniya
3. Журнал ситилинк[Электронный ресурс]/Режим доступа: https://journal.citilink.ru/articles/ot-tesnyh-podvalov-do-stilnyh-prostranstv-kak-izmenilis-kompyuternye-kluby-za-25-let/
4. Майкросойт Азур[Электронный ресурс]/Режим доступа: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/what-is-a-relational-database
5. Телеграф[Электронный ресурс]/Режим доступа: https://telegra.ph/Kak-nazyvaetsya-kompyuternyj-klub-Internet-kafe-chto-ehto-takoe-i-zachem-lyudi-hodyat-tuda-04-04#:~:text=Компьютерный%20клуб%2C%20предоставляющий%20доступ%20к,доступны%20Wi-Fi%2C%20принтеры%20и%20сканеры

Приложение 1

```
CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;
USE `mydb` ;

CREATE TABLE IF NOT EXISTS `mydb`.`users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NOT NULL,
  `hash_password` VARCHAR(255) NOT NULL,
  `email` VARCHAR(255) NULL,
  `phone_number` CHAR(13) CHARACTER SET 'ascii' NOT NULL,
  `balance` DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.00,
  `bonuce_balance` DECIMAL(10,2) UNSIGNED NOT NULL DEFAULT 0.00,
  `passport_series` CHAR(10) NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE,
  UNIQUE INDEX `email_UNIQUE` (`email` ASC) VISIBLE,
  UNIQUE INDEX `phone_number_UNIQUE` (`phone_number` ASC) VISIBLE,
  UNIQUE INDEX `passport_series_UNIQUE` (`passport_series` ASC)
VISIBLE)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`rooms` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT(5000) NULL,
  `vip` CHAR(1) NOT NULL DEFAULT 0,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`computers` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(45) NOT NULL,
  `description` TEXT(5000) NULL,
  `price_per_hours` DECIMAL(10,2) NOT NULL DEFAULT 0.00,
  `status` CHAR(1) NULL,
  `rooms_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_computers_rooms1_idx` (`rooms_id` ASC) VISIBLE,
  CONSTRAINT `fk_computers_rooms1`
    FOREIGN KEY (`rooms_id`)
    REFERENCES `mydb`.`rooms` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`employees` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `username` VARCHAR(255) NOT NULL,
  `hash_password` VARCHAR(255) NOT NULL,
  `name` VARCHAR(255) NULL,
  `surename` VARCHAR(255) NULL,
```

```
  `patronymic` VARCHAR(255) NULL,
  `description` TEXT(5000) NULL,
  `salary_per_hour` DECIMAL(10,2) NOT NULL DEFAULT 0.00,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `username_UNIQUE` (`username` ASC) VISIBLE)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`sessions` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `start_time` TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `end_time` TIMESTAMP NULL,
  `date` DATE NULL,
  `users_id` INT NOT NULL,
  `employees_id` INT NOT NULL,
  `computers_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_sessions_users_idx` (`users_id` ASC) VISIBLE,
  INDEX `fk_sessions_computers1_idx` (`computers_id` ASC) VISIBLE,
  INDEX `fk_sessions_employees1_idx` (`employees_id` ASC) VISIBLE,
  CONSTRAINT `fk_sessions_users`
    FOREIGN KEY (`users_id`)
    REFERENCES `mydb`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_sessions_computers1`
    FOREIGN KEY (`computers_id`)
    REFERENCES `mydb`.`computers` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_sessions_employees1`
    FOREIGN KEY (`employees_id`)
    REFERENCES `mydb`.`employees` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

CREATE TABLE IF NOT EXISTS `mydb`.`payments` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `replenishment` DECIMAL(10,2) NULL,
  `users_id` INT NOT NULL,
  `employees_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `fk_payments_users1_idx` (`users_id` ASC) VISIBLE,
  INDEX `fk_payments_employees1_idx` (`employees_id` ASC) VISIBLE,
  CONSTRAINT `fk_payments_users1`
    FOREIGN KEY (`users_id`)
    REFERENCES `mydb`.`users` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_payments_employees1`
    FOREIGN KEY (`employees_id`)
    REFERENCES `mydb`.`employees` (`id`)
    ON DELETE NO ACTION
```

```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Приложение 2

```
-- Вставка тестовых данных в таблицу users
INSERT INTO `mydb`.`users` (`username`, `hash_password`, `email`,
`phone_number`, `balance`, `bonuce_balance`, `passport_series`)
VALUES
('user1', 'hash1', 'user1@example.com', '1234567890123', 100.00,
10.00, '4500123456'),
('user2', 'hash2', 'user2@example.com', '2234567890123', 200.00,
20.00, '4501234567'),
('user3', 'hash3', 'user3@example.com', '3234567890123', 300.00,
30.00, '4502345678'),
('user4', 'hash4', 'user4@example.com', '4234567890123', 400.00,
40.00, '4503456789'),
('user5', 'hash5', 'user5@example.com', '5234567890123', 500.00,
50.00, '4504567890'),
('user6', 'hash6', 'user6@example.com', '6234567890123', 600.00,
60.00, '4505678901'),
('user7', 'hash7', 'user7@example.com', '7234567890123', 700.00,
70.00, '4506789012'),
('user8', 'hash8', 'user8@example.com', '8234567890123', 800.00,
80.00, '4507890123'),
('user9', 'hash9', 'user9@example.com', '9234567890123', 900.00,
90.00, '4508901234'),
('user10', 'hash10', 'user10@example.com', '0234567890123', 1000.00,
100.00, '4509012345');

-- Вставка тестовых данных для комнат в компьютерном клубе с
описанием на русском языке
INSERT INTO `mydb`.`rooms` (`title`, `description`, `vip`)
VALUES
('Комната 1', 'Это стандартная комната с высокопроизводительными
игровыми ПК.', '0'),
('Комната 2', 'Эта комната оснащена передовыми игровыми креслами и
периферией.', '0'),
('VIP Комната 1', 'Эксклюзивная VIP-комната с первоклассными
игровыми установками и приватным лаунжем.', '1'),
('Комната 3', 'Просторная комната с множеством экранов для групповых
игр.', '0'),
('VIP Комната 2', 'Премиальная VIP-комната с персональным
обслуживанием и новейшими технологиями.', '1'),
('Комната 4', 'Тихая комната для сосредоточенных игровых сессий.',
'0'),
('Комната 5', 'Комната, оборудованная новейшими технологиями
виртуальной реальности.', '0'),
('VIP Комната 3', 'Роскошная VIP-комната с индивидуально собранными
игровыми установками.', '1'),
('Комната 6', 'Комната с эргономичной мебелью и мягким освещением
для длительных игровых сессий.', '0'),
('Комната 7', 'Комната, предназначенная для тренировки в
киберспорте.', '0');

-- Вставка тестовых данных для сотрудников
```

```
INSERT INTO `mydb`.`employees` (`username`, `hash_password`, `name`,
`surename`, `patronymic`, `description`, `salary_per_hour`)
VALUES
('ivanov', 'hash1', 'Иван', 'Иванов', 'Иванович', 'Старший системный
администратор', 500.00),
('petrov', 'hash2', 'Петр', 'Петров', 'Петрович', 'Главный инженер
по информационной безопасности', 600.00),
('sidorov', 'hash3', 'Сидор', 'Сидоров', 'Сидорович', 'Менеджер по
IT-проектам', 550.00),
('smirnova', 'hash4', 'Анна', 'Смирнова', 'Игоревна', 'Системный
аналитик', 520.00),
('vasileva', 'hash5', 'Ольга', 'Васильева', 'Владимировна',
'Разработчик программного обеспечения', 580.00),
('nikolaev', 'hash6', 'Николай', 'Николаев', 'Николаевич',
'Технический директор', 650.00),
('novikova', 'hash7', 'Мария', 'Новикова', 'Алексеевна', 'Специалист
по поддержке пользователей', 450.00),
('fedorov', 'hash8', 'Федор', 'Федоров', 'Федорович', 'Сетевой
администратор', 480.00),
('alexeev', 'hash9', 'Алексей', 'Алексеев', 'Алексеевич',
'Программист', 530.00),
('kozlov', 'hash10', 'Дмитрий', 'Козлов', 'Дмитриевич', 'Веб-
разработчик', 500.00);

-- Вставка тестовых данных для компьютеров
INSERT INTO `mydb`.`computers` (`title`, `description`,
`price_per_hours`, `status`, `rooms_id`)
SELECT
  CONCAT('Компьютер ', c.id) AS title,
  CONCAT('Игровой компьютер номер ', c.id) AS description,
  100.00 AS price_per_hours,
  '1' AS status,
  r.id AS rooms_id
FROM `mydb`.`rooms` r
JOIN (
    SELECT id, title
    FROM `mydb`.`rooms`
    ORDER BY id
) AS c ON r.title = c.title;

-- Вставка данных в таблицу sessions с использованием JOIN
INSERT INTO `mydb`.`sessions` (`start_time`, `end_time`, `date`,
`users_id`, `employees_id`, `computers_id`)
SELECT
  NOW() AS start_time,
  NULL AS end_time,
  CURDATE() AS date,
  u.id AS users_id,
  e.id AS employees_id,
  c.id AS computers_id
FROM `mydb`.`users` u
JOIN (
    SELECT id
```

```
    FROM `mydb`.`employees`
    ORDER BY RAND() -- пример случайного выбора сотрудника
    LIMIT 1
) AS e ON 1=1 -- просто для соответствия условию JOIN
JOIN (
    SELECT id
    FROM `mydb`.`computers`
    ORDER BY RAND() -- пример случайного выбора компьютера
    LIMIT 1
) AS c ON 1=1; -- просто для соответствия условию JOIN

-- Вставка тестовых данных в таблицу payments
INSERT INTO `mydb`.`payments` (`replenishment`, `users_id`,
`employees_id`)
SELECT
  ROUND(RAND() * 1000, 2) AS replenishment,
  u.id AS users_id,
  e.id AS employees_id
FROM `mydb`.`users` u
JOIN (
    SELECT id
    FROM `mydb`.`employees`
    ORDER BY RAND() -- пример случайного выбора сотрудника
    LIMIT 1
) AS e ON 1=1 -- просто для соответствия условию JOIN;
```

Приложение 3

```
DELIMITER $$

CREATE FUNCTION `mydb`.`get_total_balance` (userId INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE totalBalance DECIMAL(10,2);
    DECLARE userBalance DECIMAL(10,2);
    DECLARE userBonusBalance DECIMAL(10,2);

    -- Получаем текущий баланс пользователя
    SELECT balance, bonuce_balance INTO userBalance,
userBonusBalance
    FROM `mydb`.`users`
    WHERE id = userId;

    -- Рассчитываем общий баланс
    SET totalBalance = userBalance + userBonusBalance;

    -- Возвращаем общий баланс
    RETURN totalBalance;
END$$
```

Приложение 4

```
CREATE VIEW `mydb`.`user_sessions` AS
SELECT
    u.id AS user_id,
    u.username,
    u.email,
    u.phone_number,
    u.balance,
    u.bonuce_balance,
    s.id AS session_id,
    s.start_time,
    s.end_time,
    s.date,
    e.id AS employee_id,
    e.username AS employee_username,
    c.id AS computer_id,
    c.title AS computer_title,
    c.price_per_hours AS computer_price_per_hour
FROM
    `mydb`.`users` u
JOIN
    `mydb`.`sessions` s ON u.id = s.users_id
JOIN
    `mydb`.`employees` e ON s.employees_id = e.id
JOIN
    `mydb`.`computers` c ON s.computers_id = c.id;
```

Приложение 5

```
DELIMITER $$

CREATE PROCEDURE AddPayment(IN userId INT, IN employeeId INT, IN
amount DECIMAL(10,2))
BEGIN
    DECLARE currentBalance DECIMAL(10,2);
    DECLARE maxLimit DECIMAL(10,2) DEFAULT 10000.00; --
Устанавливаем лимит пополнения

    -- Начало транзакции
    START TRANSACTION;

    -- Получаем текущий баланс пользователя
    SELECT balance INTO currentBalance FROM `mydb`.`users` WHERE
id = userId FOR UPDATE;

    -- Проверяем, не превышает ли пополнение лимит
    IF currentBalance + amount <= maxLimit THEN
        -- Вставляем запись в таблицу payments
        INSERT INTO `mydb`.`payments` (`replenishment`,
`users_id`, `employees_id`)
        VALUES (amount, userId, employeeId);

        -- Фиксируем транзакцию
        COMMIT;
    ELSE
        -- Откатываем транзакцию, если условие не выполнено
        ROLLBACK;
    END IF;
END$$

DELIMITER ;
```

Приложение 6

```
DELIMITER $$

CREATE PROCEDURE CreateSession(
    IN userId INT,
    IN employeeId INT,
    IN computerId INT,
    IN sessionDurationHours DECIMAL(5,2)
)
BEGIN
    DECLARE computerPricePerHour DECIMAL(10,2);
    DECLARE userBalance DECIMAL(10,2);
    DECLARE userBonusBalance DECIMAL(10,2);
    DECLARE totalCost DECIMAL(10,2);

    -- Начало транзакции
    START TRANSACTION;

    -- Получаем стоимость компьютера в час
    SELECT price_per_hours INTO computerPricePerHour
    FROM `mydb`.`computers`
    WHERE id = computerId
    FOR UPDATE;

    -- Получаем текущий баланс пользователя
    SELECT balance, bonuce_balance INTO userBalance,
userBonusBalance
    FROM `mydb`.`users`
    WHERE id = userId
    FOR UPDATE;

    -- Рассчитываем общую стоимость сессии
    SET totalCost = computerPricePerHour * sessionDurationHours;

    -- Проверяем, достаточно ли средств на счету пользователя
    IF (userBalance + userBonusBalance) >= totalCost THEN
        -- Вставляем новую запись в таблицу sessions
        INSERT INTO `mydb`.`sessions` (`start_time`, `end_time`,
`date`, `users_id`, `employees_id`, `computers_id`)
        VALUES (NOW(), NOW() + INTERVAL sessionDurationHours HOUR,
CURDATE(), userId, employeeId, computerId);

        -- Сначала списываем с основного баланса
        IF userBalance >= totalCost THEN
            UPDATE `mydb`.`users`
            SET balance = balance - totalCost
            WHERE id = userId;
        ELSE
            -- Списываем с основного баланса все, что можно, и
остаток с бонусного баланса
            UPDATE `mydb`.`users`
            SET balance = 0,
```

```
                bonuce_balance = bonuce_balance - (totalCost -
userBalance)
            WHERE id = userId;
        END IF;

        -- Фиксируем транзакцию
        COMMIT;
    ELSE
        -- Откатываем транзакцию, если недостаточно средств
        ROLLBACK;
    END IF;
END$$

DELIMITER ;
```

Приложение 7

QR код на репозиторий в git-hub

Ссылка для ручного ввода:

https://github.com/skeletus-design/course_work

Отчет о проверке № 8897543

Дата выгрузки: 2024-06-20 07:48:35
Пользователь: nikita.voro24nin@gmail.com, ID: 8897543

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 8897543
Имя исходного файла: Курсовая.docx
Размер файла: 0.33 МБ
Размер текста: 44243
Слов в тексте: 6300
Число предложений: 711

Информация об отчете

Дата: 2024-06-20 07:48:35 - Последний готовый отчет
Оценка оригинальности: 89%
Заимствования: 11%

89.17% 10.83%

Источники:

Доля в тексте	Ссылка
48.50%	https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B7%D0%BD%D0%B5%D1%...
23.50%	https://scienceforum.ru/2020/article/2018021900
8.05%	https://telegra.ph/Kak-nazyvaetsya-kompyuternyj-klub-Internet-ka...
7.20%	https://www.jetinfo.ru/metody-i-sredstva-modelirovaniya-biznes-p...

Информация о документе:

Частное учреждение профессионального образования "Высшая школа предпринимательства" (ЧУПО "ВШП") КУРСОВОЙ ПРОЕКТ "Разработка базы данных для компьютерного клуба" Выполнил: студент 3-го курса специальности 09.02.07 "Информационные системы и программирование" Воронин Никита Валерьевич подпись: _____ Проверил: преподаватель дисциплины, преподаватель ЧУПО "ВШП", к.ф.н. Ткачев П.С. оценка: _____ подпись: _____ Оглавление Введение 3 Первая глава 6 Определение бизнес процессов условного компьютерного клуба 6 Формулировка требований к разрабатываемой базе данных 8 Выбор СУБД для реализации базы данных 10 Краткий вывод после проведения анализа возможных решений и определения бизнес процессов условного компьютерного клуба "Игрульки" 14 Вторая глава 15 Построение схемы базы данных 15 Возможности и использование MySQL Workbench 15 ER диаграмма и набор таблиц проекта 16 Реализация бизнес процессов на уровне СУБД 25 Заключение 29 Список используемой литературы 30 Электронные ресурсы 30 Приложение 1 31 33 Приложение 2 34 36 Приложение 3 37 Приложение 4 38 38 Приложение 5 39 Приложение 6 40 41 Приложение 7 42 Введение Актуальность Сейчас игровая индустрия стремительно развивается и не все люди в силу некоторых личных обстоятельств могут позволить себе дорогие комплектующие или консоли, по этому они, время от времени, посещают компьютерные клубы. Данная тема может считаться актуальной и достаточно интересной, по скольку многие процессы компьютерных клубов можно переложить также на другой бизнес или другие концепции. Определение цели работы Для начала следует внести некоторую ясность в тему и сформулировать цель курсовой работы. Тема курсовой работы "Разработка базы данных для компьютерного клуба". Соответственно, цель работы - разработать базу данных, отражающую некоторые бизнес процессы, для условного компьютерного клуба. Прежде чем перейти к формулировке задач, необходимых для достижения поставленной цели, стоит кратко изложить что из себя вообще представляет "компьютерный клуб" как сущность или явление. По версии сайта

telegra.ph[5], компьютерные клубы (также известные как интернет-кафе или киберкафе) - это заведение предоставляющее доступ к Интернету. Это публичное заведение, где люди могут приходить работать, играть в онлайн-игры, общаться в социальных сетях и просто проводить время в виртуальном мире. Внутри интернет-кафе обычно имеются несколько компьютерных столов, каждый со своей клавиатурой, мышью и монитором. Кроме того, в заведении часто бывают доступны Wi-Fi, принтеры и сканеры. Из этого определения уже можно понять что-то о возможных бизнес процессах в условном компьютерном клубе, но это лучше оставить для основного раздела работы. Краткая история возникновения феномена в России В России первые компьютерные клубы появились в 1996 году: легендарные "Орки" в Москве и "Виртус" в Санкт-Петербурге. Редкие везунчики, у которых были компьютеры, ходили друг к другу с собственными машинами, чтобы поиграть по локалке. Все остальные тусовались в игровых клубах. "Только откроется клуб на районе, как в школе уже говорят об этом: "Ну че, ходил в новые компы?" Мы с друзьями неделю экономили на обедах, чтобы на выходных посидеть в компах подольше. Часто приходишь в компы, стучишь в дверь и через решетку спрашиваешь у админа: "Есть свободные?" Он отвечает, что через два часа освободятся два компа. Забиваешь очередь и уходишь ждать. Люди стояли на морозе, просили пустить в тамбур погреться или в зал - посмотреть за игрой других. Еще бывало так: на ночь собиралась компания, админ запирал их всех внутри и уходил домой спать. Жуткая вещь: на окнах решетки, телефонов нет никаких, если вдруг пожар - хана." Алексей Офицеров, заядлый посетитель клубов в 90-х [3] Современные же компьютерные клубы отличаются следующими особенностями: 1. Мощное оборудование. Современные клубы устанавливают сетевое оборудование и игровые системы, которые полностью удовлетворяют потребности популярных проектов и имеют запас мощности на несколько лет вперёд. 2. Пополнение списка актуальных игр. Современные клубы следят за выходом дополнений, оперативной установкой патчей и фиксов. 3. Удобство клиентов. Клубы предоставляют эргономичные девайсы для игровой зоны, а также разрешают приходить со своими. 4. Сервис. Современные клубы стремятся к безупречному сервису, например, имеют удобные парковочные места, комнаты для отдыха и lounge-зоны. Разница между "было" и "стало" вполне очевидна. Постановка задач Исходя из определения выше - сформулируем возможные задачи для достижения цели курсовой работы: 1. Определение бизнес процессов условного компьютерного клуба. 2. Сформулировать требования к разрабатываемой базе данных 3. Выбор СУБД(системы управления базами данных) для реализации базы данных. 4. Построение схемы базы данных. 5. Разработать набор связанных таблиц базы данных, исходя из требований к базе данных и бизнес процессов условного компьютерного клуба. 6. Заполнить таблицы тестовыми данными. 7. Реализовать бизнес процессы на уровне разработанной базы данных средствами выбранной СУБД. 8. Провести тестирование реализованных процессов тестовыми данными. Объект исследования Объектом исследования является процесс разработки баз данных, предназначенной для обслуживания бизнес процессов условного компьютерного клуба. Метод исследования Моделирование, а именно моделирование процессов внутри компьютерного клуба и пользовательского поведения. Первая глава Определение бизнес процессов условного компьютерного клуба Для наглядности нужно дать определение "бизнес процессу", как понятию. Сам по себе бизнес процесс - совокупность взаимосвязанных мероприятий или работ, направленных на создание определённого продукта или услуги для потребителей. [1] Выделяют три вида бизнес-процессов: 1. управляющие - бизнес-процессы, которые управляют функционированием системы, такие как корпоративное управление и стратегический менеджмент; 2. операционные - представляющие основную деятельность организации, создающие основной поток доходов (снабжение, производство, маркетинг, продажи или взыскание долгов); 3. поддерживающие - обслуживающие организацию (бухгалтерский учет, подбор персонала, техническая поддержка). Бизнес-процесс начинается со спроса потребителя и заканчивается его удовлетворением. Процессно-ориентированные организации стараются устранять барьеры и задержки, возникающие на стыке двух различных подразделений организации при выполнении одного бизнес-процесса. Бизнес-процесс может быть декомпозирован на несколько подпроцессов, процедур и функций, которые имеют собственные атрибуты, однако также направлены на достижение цели основного бизнес-процесса. Такой анализ бизнес-процессов обычно включает в себя составление карты бизнес-процесса и его подпроцессов, разнесенных между определёнными уровнями активности. Бизнес-процессы должны быть построены таким образом, чтобы создавать стоимость и ценность для потребителей и исключать любые необязательные или вовсе лишние активности. На выходе правильно построенных бизнес-процессов увеличиваются ценность для потребителя и рентабельность (меньшая себестоимость производства товара или услуги). Бизнес-процессы могут подвергаться различному анализу в зависимости от целей моделирования. Анализ бизнес-процессов может применяться при бизнес-моделировании, функционально-стоимостном анализе, формировании организационной структуры, реинжиниринге бизнес-процессов, автоматизации технологических процессов. В данном случае условный компьютерный клуб, назовем его "Игрульки", является перспективно масштабируемым бизнесом с возможностью расширения. Относительно этого процессы внутри компьютерного клуба должны быть универсальными для всех филиалов организации, быстро и точно обрабатывать данные пользователей и сотрудников, работать надежно и бесперебойно. В основе клуб должен приносить прибыль за счет посетителей, которые пополняют кошелек на своем аккаунте для запуска сессии на выбранном компьютере. Тезисно представленные в виде списка искомые бизнес процессы: 1. Хранение и обслуживание пользовательских данных клиентов и сотрудников клуба "Игрульки" 2. Реализация функционала пополнения баланса пользовательского аккаунта. 3. Активация сессий пользователя за н-ным компьютером в н-ном зале при н-ном сотруднике на смене и достаточном балансе. 4. Сохранение и отображение данных о совершенных пополнениях. Необходимые основные процессы сформулированы и оглашены. Исходя из этого можно перейти к формулировке требований к базе данных. Формулировка требований к разрабатываемой базе данных Какие вообще бывают требования к базам данных? Согласно источнику[2], существуют следующие виды бизнес-требований: 1. Функциональные. Описывают конкретные функции системы или проекта, которые больше всего важны для бизнеса. Отвечают на вопрос "Что должна делать система?". 2. Нефункциональные. Описывают характеристики продукта, которые определяют, как он должен выполнять свои функции. Отвечают на вопрос "Как должна работать функция системы?". Бизнес-требования помогают: 1. зафиксировать требования заказчика проекта; 2. собрать потребности клиентов; 3. команде разработчиков и всем

сотрудникам понять, какой конечный продукт нужно получить; Исходя из изложенного выше можно выделить несколько требований: 1. База данных должна быть реляционной, по скольку в реляционной базе данные организованы в таблицы, содержащие информацию о каждом объекте и представляющие заранее определённые категории через строки и столбцы. Такое структурирование данных делает доступ к ним эффективным и гибким (Что такое реляционные базы данных [4]) 2. База данных должна содержать таблицы для организации работы сотрудников, пользовательского интерфейса и некоторой логики клуба "Игрульки" 3. Таблицы должны быть связаны между собой для организации логики бизнес процессов и корректного отображения данных. 4. База данных должна поддрерживать большое количество асинхронных запросов со стороны клиентов сервиса, следовательно, должна быть легко внедряемой и универсальной. Подробнее поговорим о реляционных базах данных и какие им бывают альтернативы: Реляционные базы данных предназначены для поддержания согласованности и целостности данных, а также обеспечения связей и ограничений между различными таблицами. Они полагаются на язык структурированных запросов (SQL) для запроса, манипулирования и организации данных. Альтернатива реляционным базам данных - нереляционные базы данных (базы данных NoSQL). Они предназначены для хранения данных в форматах, отличных от таблиц, и предоставляют более простое, гибкое и масштабируемое решение для хранения и управления неструктурированными или полуструктурированными данными. Некоторые широко используемые нереляционные базы данных включают: - MongoDB, - Cassandra, - Redis, - Neo4j. Однако, данные СУБД не совсем удовлетворяют все критерии для реализации сервиса условного компьютерного клуба "Игрульки", по этому от них в последствии придется отказаться. Основные требования к базе данных, на мой взгляд, сформулированы. Исходя из этих требований можно перейти к следующей задаче. Выбор СУБД для реализации базы данных В качестве системы управления базами данных была выбрана MySQL. Данная СУБД является реляционной, достаточно производительной и функциональной для выполнения поставленных выше задач и требований, а именно: 1. MySQL имеет в своем функционале различные функции безопасности, включая возможность установки привилегий пользователя, шифрование данных, аутентификацию и аудит, чтобы обезопасить информацию, хранящуюся в базах данных. Поддерживает множество языков программирования, таких как: Python, C/C++, Java, JavaScript, Go, Delphi, Erlang и др. 2. MySQL обладает хорошей производительностью и быстродействием благодаря оптимизированным алгоритмам выполнения запросов. Умеет работать с большими объемами данных с минимальными задержками. Легко масштабируется и подходит для больших баз данных. 3. MySQL может использоваться как для небольших веб-приложений, так и для серьезных корпоративных систем. СУБД предлагает различные методы масштабирования, включая горизонтальное и вертикальное масштабирование, что дает возможность расширять базы данных при возникновении необходимости. Это позволяет обрабатывать большое количество одновременных запросов и поддерживать беспрепятственный доступ к данным для пользователей. 4. В MySQL реализованы различные типы данных, индексы, хранимые процедуры, триггеры и другие функции, обеспечивающие гибкость при обработке данных. Это дает возможность создания сложных схем данных. MySQL совместима с довольно большим количеством операционных системам. 5. Бесплатность и открытые исходники: рассматриваемый продукт имеет открытый исходный код и может быть использован бесплатно. Это делает его доступным для большого круга разработчиков и пользователей. 6. MySQL имеет большое сообщество разработчиков и пользователей, что дает возможность доступа к различным ресурсам и помощи при возникновении проблем. Перечисленные возможности как нельзя лучше подходят для реализации подобной базы данных. Однако, ниже приведены некоторые недостатки MySQL, на которые стоит обратить внимание: 1. MySQL может иметь проблемы с производительностью при обработке больших объемов данных или когда требуется обработка сложных запросов или подключение большого количества клиентов. Это может привести к замедлению работы или даже отказу в обслуживании. 2. MySQL имеет ограниченный набор типов данных по сравнению с некоторыми другими СУБД. Например, поддержка временных данных и географических данных ограничена по сравнению с некоторыми другими СУБД. 3. MySQL может столкнуться с ограничениями в масштабируемости и доступности при взаимодействии с большим количеством клиентов. Тогда отказоустойчивость и возможность горизонтального масштабирования могут быть ограничены. 4. MySQL может быть сложным для администрирования, особенно для новичков. Настройка и оптимизация параметров конфигурации может быть сложной задачей, требующей глубоких знаний и опыта. 5. MySQL может иметь уязвимости в сфере безопасности, например, возможность атаки SQL-инъекцией или недостаточная защита данных. Это требует дополнительных мер безопасности для уверенной надежности и защиты баз данных. Также, для большей информативности нужно выделить области применения MySQL: 1. MySQL часто применяется для построения динамических сайтов и интернет-приложений. При этом СУБД используется в сочетании с веб-серверами (Apache, Nginx) 2. Многие предприятия организации используют MySQL для создания и управления баз(ами) данных своих бизнес-приложений. Данная СУБД дает возможность надежно хранить и легко получать доступ к данным, что позволяет эффективно работать с клиентами, заказами, продуктами и другими составляющими бизнеса, получая информацию о них из корпоративных баз данных. 3. Базы MySQL используются как хранилище данных для аналитических и отчетных систем. С помощью рассматриваемого программного продукта можно хранить большие объемы данных, а также выполнять сложные запросы и агрегировать данные, чтобы проводить анализ и создавать отчеты. 4. MySQL используется также как СУБД для мобильных приложений, хранящих различные типы данных, такие как пользовательские профили, настройки, результаты и др. Во многом из-за своей надежности и хорошей производительности MySQL хорошо подходит для решения данных задач. 5. MySQL может быть встроен в другие приложения и устройства для взаимодействия с небольшими базами данных. Это могут быть системы управления контентом (CMS), блоги, форумы, физические устройства, такие как маршрутизаторы и т. д. Почему MySQL а не PostGres ? Выбор между MySQL и PostgreSQL зависит от конкретных потребностей и приоритетов бизнеса. Обе СУБД имеют свои сильные стороны и особенности. В контексте компьютерного клуба, MySQL может быть предпочтительным по следующим причинам: 1. Легкая установка и настройка, что особенно важно для небольших команд или тех, кто не имеет глубокого опыта в администрировании баз данных. 2. Менее сложный процесс конфигурации по сравнению с PostgreSQL. 3. Обычно MySQL превосходит PostgreSQL в операциях на чтение, что может быть важным для приложений, где чтение данных

происходит чаще, чем запись. 4. MySQL широко поддерживается многими хостинг-провайдерами и платформами. 5. Легкая интеграция MySQL с популярными веб-приложениями, такими как WordPress, Joomla и другие, что может быть полезно для создания веб-сайта клуба или онлайн-сервисов. 6. Более интуитивно понятный интерфейс и множество инструментов для администрирования, таких как phpMyAdmin. 7. Оптимизирован для работы с веб-приложениями и часто используется в популярных веб-технологиях. 8. Высокая производительность в типичных веб-нагрузках, что важно для веб-сайтов компьютерного клуба. 9. Обычно MySQL требует меньше усилий на администрирование и техническую поддержку, что может быть важным фактором для небольших организаций с ограниченными ресурсами. 10. Большое количество специалистов, знакомых с MySQL, что облегчает поиск сотрудников для технической поддержки и развития. Примеры использования в компьютерном клубе 1. Регистрация и учет клиентов: Быстрая обработка регистрационных данных и хранение истории посещений клиентов. 2. Управление оборудованием: Учет